UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/069,805	11/07/2002	Gilbert Wolrich	10559-307US1	7118

20985 7590 04/08/2008
FISH & RICHARDSON, PC
P.O. BOX 1022
MINNEAPOLIS, MN 55440-1022

EXAMINER	
HUISMAN, DAVID J	
ART UNIT	PAPER NUMBER
2183	
MAIL DATE	DELIVERY MODE
04/08/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/069,805	WOLRICH ET AL.	
	Examiner	Art Unit	
	DAVID J. HUISMAN	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 January 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2,4 and 7-15* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,2,4 and 7-15* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *14 September 2006* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *1/15/2008*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-2, 4, and 7-15 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: IDS as received on 1/15/2008, and Amendment and Extension of Time as received on 1/22/2008.

Specification

3. The specification is objected to as failing to provide proper antecedent basis for the claimed subject matter. See 37 CFR 1.75(d)(1) and MPEP § 608.01(o). Correction of the following is required: The specification does not include disclosure of a "computer program product" or a "computer readable storage medium".

Maintained Rejections

4. Applicant has failed to overcome the prior art rejections set forth in the previous Office Action. Consequently, these rejections are respectfully maintained by the examiner and are copied below for applicant's convenience.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1 and 7-15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Saulsbury et al., U.S. Patent No. 6,314,510 (herein referred to as Saulsbury).

7. Referring to claim 1, Saulsbury has taught a computer program product residing on a computer readable storage medium (Fig.1, component 102; column 2, lines 42-45) comprising instructions (Fig.4), including a context branch instruction (Fig.4 and column 4, lines 44-45; note the "branch on zero" (bz) instruction) that, when executed, causes a data processing apparatus to select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (Fig.4; note that the "bz next1" instruction will branch to the target instruction associated with label "next1", which is the "btst wr1, db1" instruction) and an instruction following the context branch instruction ("st wr0, [addr2]") based on a comparison of a current number to a number specified by the context branch instruction. The bz instruction is a branch on zero instruction, which means that if the number specified by the branch instruction (zero) matches a current number (i.e., the value to be compared to zero), then a branch will occur. In this program, the "bz next1" instruction checks to see if the current number (which may be interpreted as either dirty bit db0 or the flag which would be checked by the bz instruction) is equal to 0, the specified number. See column 4, lines 39-45. It should be noted that the branch is a context branch as the branch is associated with executing contexts. See the title, abstract, and column 2, lines 40-41.

b) retrieving the selected other instruction. As is known, with a condition branch, as shown in Fig.4, either the target instruction or the subsequent instruction will be retrieved.

c) despite teaching multiple contexts, Saulsbury has not explicitly taught that the data processing apparatus is a multi-threaded data processing apparatus and that the current number is a current executing thread number and that the specified number is a thread number specified by the branch instruction. However, Official Notice is taken that multithreading and its advantages are well known and accepted in the art. Specifically, a multithreaded system executes multiple independent groups of instructions called threads. By having multiple independent groups of instructions as opposed to a single serial stream of instructions, a multithreaded system is able to switch to another thread when one thread experiences a delay. This ensures that processor resources are kept as busy as possible, which in turn increases throughput. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Saulsbury's system such that it supported multithreading. It would then follow that the aforementioned branch instruction would be executed on the multithreaded system and the current number would be the current executing thread number (i.e., a number associated with the current thread), and the thread number specified by the branch instruction would still be zero, as described above, because zero is a thread number (i.e., a number associated with a thread). It should be noted that it would be obvious to have this instruction in any multithreaded machine as "branch on zero", as it is a very well-known instruction in the art.

8. Referring to claim 7, Saulsbury has taught a method of operating a processor comprising: a) performing a comparison of a number to a number specified by a context branch instruction. See the "bz next1" instruction of Fig.4 and column 4, lines 44-45. This instruction evaluates an executing context number (dirty bit db0 or the flag set in testing db0) to see if it equals the context number specified by the branch instruction, i.e., 0, since it is a "branch if 0" instruction.

See column 4, lines 39-45. Note that the branch is a context branch as the branch is associated with contexts (i.e., executing environments). See the title, abstract, and column 2, lines 40-41.

b) selecting another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (Fig.4; note that the "bz next1" instruction will branch to the target instruction associated with label "next1", which is the "btst wr1, db1" instruction) and an instruction following the context branch instruction ("st wr0, [addr2]") based on a comparison (if the comparison yields that the numbers match, the target instruction will be selected; otherwise the following instruction will be selected).

c) retrieving the selected other instruction. As is known, with a condition branch, as shown in Fig.4, either the target instruction or the subsequent instruction will be retrieved.

d) despite teaching multiple contexts, Saulsbury has not explicitly taught that the processor is a multi-threaded processor and that the number is a thread number of an executing thread and that the specified number is a thread number specified by the branch instruction. However, Official Notice is taken that multithreading and its advantages are well known and accepted in the art. Specifically, a multithreaded system executes multiple independent groups of instructions called threads. By having multiple independent groups of instructions as opposed to a single serial stream of instructions, a multithreaded system is able to switch to another thread when one thread experiences a delay. This ensures that processor resources are kept as busy as possible, which in turn increases throughput. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Saulsbury's system such that it supported multithreading. It would then follow that the aforementioned branch instruction would be executed on the multithreaded system and the number would be the thread number of

an executing thread (i.e., a number associated with the current thread), and the thread number specified by the branch instruction would still be zero, as described above, because zero is a thread number (i.e., a number associated with a thread). It should be noted that it would be obvious to have this instruction in any multithreaded machine as "branch on zero", as it is a very well-known instruction in the art.

9. Referring to claim 8, Saulsbury, as modified, has taught a method as described in claim 7. Saulsbury has further taught selecting comprises selecting the branch target instruction if the executing context number matches the specified thread number. As discussed above, and as shown in Fig.4 and column 4, lines 39-45, if the dirty bit db0 is evaluated (by way of flag) and determined to be equal to 0, which is the condition specified by the bz instruction, then a branch will occur.

10. Referring to claim 9, Saulsbury, as modified, has taught a method as described in claim 7. Saulsbury has further taught that the thread number has valid values of 0, 1, 2, or 3. Again, see the bz instruction of Fig.4. Since the bz instruction is "branch if 0", 0 is a valid specified thread number.

11. Referring to claim 10:

a) Saulsbury has taught a processor that can execute multiple contexts, but has not explicitly taught a processor that can execute multiple threads. However, Official Notice is taken that multithreading and its advantages are well known and accepted in the art. Specifically, a multithreaded system executes multiple independent groups of instructions called threads. By having multiple independent groups of instructions as opposed to a single serial stream of instructions, a multithreaded system is able to switch to another thread when one thread

experiences a delay. This ensures that processor resources are kept as busy as possible, which in turn increases throughput. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Saulsbury's system such that it supported multithreading.

b) Saulsbury, as modified, has further taught that the processor comprises:

b1) a register stack. See Fig.1, component 110.

b2) Saulsbury has taught an execution unit (Fig.1, component 108) coupled to the register stack and a program control store that stores a context branch instruction (Fig.1, component 102, and column 2, lines 42-45) that causes the processor to:

- perform a comparison of a thread number of an executing thread to a thread number specified by the branch instruction. See the "bz next1" instruction of Fig.4 and column 4, lines 44-45. This instruction evaluates an executing thread number (dirty bit db0) to see if it equals the context number specified by the branch instruction, i.e., 0, since it is a "branch if 0" instruction. See column 4, lines 39-45. Note that the branch is a context branch as the branch is associated with contexts (i.e., executing environments). See the title, abstract, and column 2, lines 40-41. Also, the numbers are thread numbers because they are numbers associated with threads.
- select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (Fig.4; note that the "bz next1" instruction will branch to the target instruction associated with label "next1", which is the "btst wr1, db1" instruction) and an

instruction following the context branch instruction ("st wr0, [addr2]") based on a comparison (if the comparison yields that the numbers match, the target instruction will be selected; otherwise the following instruction will be selected).

- retrieve the selected other instruction. As is known, with a condition branch, as shown in Fig.4, either the target instruction or the subsequent instruction will be retrieved.

b3) Saulsbury has not explicitly taught a program counter for each executing thread. However, this limitation is deemed to be inherent by the examiner. A program counter must exist because the program counter is the component that holds the address of the next instruction to be fetched. Without the PC, the system would not be able to fetch an instruction, which means that no work would be done. A program counter has to exist for each thread because each thread includes instructions that need to be fetched and executed.

b4) Saulsbury has not taught that the execution unit is an arithmetic logic unit. However, Official Notice is taken that ALUs and their advantages are well known execution units that are expected in the art. An ALU is the term given to a unit that is able to perform arithmetic operations (addition, subtraction, etc.) and logical operations (AND, OR, NOT, etc.). Clearly, by implementing an ALU, the system would be able to perform a variety of operations on data. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify the execution unit of Saulsbury to be an ALU so that Saulsbury is able to perform both arithmetic and logical operations.

12. Referring to claim 11, Saulsbury, as modified, has taught a processor as described in claim 10. Saulsbury has further taught that the context branch instruction causes the processor to select the branch target instruction if the executing thread number matches the specified thread number. As discussed above, and as shown in Fig.4 and column 4, lines 39-45, if the dirty bit db0, or the flag set in testing db0, is evaluated and determined to indicate 0, which is the condition specified by the bz instruction, then a branch will occur.

13. Referring to claim 12, Saulsbury, as modified, has taught a processor as described in claim 10. Saulsbury has further taught that the thread number has valid values of 0, 1, 2, or 3. Again, see the bz instruction of Fig.4. Since the bz instruction is "branch if 0", 0 is a valid specified context number.

14. Referring to claim 13, Saulsbury has taught a computer program product residing on a computer-readable storage medium (Fig.1, component 102; column 2, lines 42-45), for causing a processor to perform a function, comprises instructions (Fig.4) causing the processor to:
a) perform a comparison of a number to a number specified by a branch instruction. See the "bz next1" instruction of Fig.4 and column 4, lines 44-45. This instruction evaluates a number (dirty bit db0 or the flag set in testing db0) to see if it equals the number specified by the branch instruction, i.e., 0, since it is a "branch if 0" instruction. See column 4, lines 39-45. Note that the branch is a context branch as the branch is associated with contexts (i.e., executing environments). See the title, abstract, and column 2, lines 40-41.
b) select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (Fig.4; note that the "bz next1" instruction will branch to the target instruction associated with label "next1", which is the "btst

wr1, db1" instruction) and an instruction following the context branch instruction ("st wr0, [addr2]") based on a comparison (if the comparison yields that the numbers match, the target instruction will be selected; otherwise the following instruction will be selected).

c) retrieve the selected other instruction. As is known, with a condition branch, as shown in Fig.4, either the target instruction or the subsequent instruction will be retrieved.

d) despite teaching multiple contexts, Saulsbury has not explicitly taught that the processor executes multiple threads and that the number is a thread number of an executing thread and that the specified number is a thread number specified by the branch instruction. However, Official Notice is taken that multithreading and its advantages are well known and accepted in the art. Specifically, a multithreaded system executes multiple independent groups of instructions called threads. By having multiple independent groups of instructions as opposed to a single serial stream of instructions, a multithreaded system is able to switch to another thread when one thread experiences a delay. This ensures that processor resources are kept as busy as possible, which in turn increases throughput. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Saulsbury's system such that it supported multithreading. It would then follow that the aforementioned branch instruction would be executed on the multithreaded system and the number would be the thread number of an executing thread (i.e., a number associated with the current thread), and the thread number specified by the branch instruction would still be zero, as described above, because zero is a thread number (i.e., a number associated with a thread). It should be noted that it would be obvious to have this instruction in any multithreaded machine as "branch on zero", as it is a very well-known instruction in the art.

15. Referring to claim 14, Saulsbury, as modified, has taught a product as described in claim 13. Saulsbury has further taught that the context branch instruction causes the processor to select the branch target instruction if the executing thread number matches the specified thread number. As discussed above, and as shown in Fig.4 and column 4, lines 39-45, if the dirty bit db0, by way of a zero-flag, is evaluated and determined to be equal to 0, which is the condition specified by the bz instruction, then a branch will occur.

16. Referring to claim 15, Saulsbury, as modified, has taught a product as described in claim 13. Saulsbury has further taught that the thread number has valid values of 0, 1, 2, or 3. Again, see the bz instruction of Fig.4. Since the bz instruction is "branch if 0", 0 is a valid specified thread number.

17. Claims 1, 7, 10, and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Steven et al., "ALU Design and Processor Branch Architecture," *Microprocessing and Microprogramming*, 1993 (as cited by applicant and herein referred to as Steven).

18. Referring to claims 1, 7, and 13, Steven has taught a computer program product residing on a computer readable storage medium comprising instructions (this is inherent as instructions must be stored somewhere), including a context branch instruction (see page 268, section 4.4, and note the Bcc instruction) that, when executed, causes a data processing apparatus to select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (page 268; note that the Bcc instruction will branch to the target instruction associated with the label) and an instruction following the context branch instruction (the fall-through instruction, which inherent exists in conditional

branching) based on a comparison of a current number to a number specified by the context branch instruction. The Bcc instruction compares a register value (specified number) to a current number (either another register value or immediate specified by the branch instruction). It should be noted that the branch is a context branch as the branch is numbers are associated with executing contexts (i.e., environments).

b) retrieving the selected other instruction. As is known, with a condition branch such as the Bcc instruction, either the target instruction or the subsequent instruction will be retrieved.

c) Steven has not explicitly taught a multithreaded implementation and that the current number is a current executing thread number and that the specified number is a thread number specified by the branch instruction. However, Official Notice is taken that multithreading and its advantages are well known and accepted in the art. Specifically, a multithreaded system executes multiple independent groups of instructions called threads. By having multiple independent groups of instructions as opposed to a single serial stream of instructions, a multithreaded system is able to switch to another thread when one thread experiences a delay. This ensures that processor resources are kept as busy as possible, which in turn increases throughput. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Steven's system such that it supports multithreading. It would then follow that the aforementioned branch instruction would be executed on the multithreaded system and the current number would be the current executing thread number (i.e., a number associated with the current thread), and the number specified by the branch instruction would be a specified thread number as it would be associated with a thread. It should be noted that it would be obvious to

have this instruction in any multithreaded machine as "branch on zero", as it is a very well-known instruction in the art.

19. Referring to claim 10, Steven has taught a processor but has not explicitly taught a processor that can execute multiple threads. However, Official Notice is taken that multithreading and its advantages are well known and accepted in the art. Specifically, a multithreaded system executes multiple independent groups of instructions called threads. By having multiple independent groups of instructions as opposed to a single serial stream of instructions, a multithreaded system is able to switch to another thread when one thread experiences a delay. This ensures that processor resources are kept as busy as possible, which in turn increases throughput. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Steven's system such that it supports multithreading.

b) Steven, as modified, has further taught that the processor comprises:

b1) a register stack. See page 268, and note that since registers are specified by instructions, a register stack exists.

b2) an arithmetic logic unit (See the ALU of Fig.8) coupled to the register stack and a program control store that stores a context branch instruction (this is deemed inherent as a memory must exist to store instructions) that causes the processor to:

- perform a comparison of a context number of an executing context to a context number specified by the branch instruction. See section 4.4 on page 268. The Bcc instruction compares a register value (specified context number) to a current executing context number (either another register value or immediate specified by

the branch instruction). It should be noted that the branch is a context branch and the numbers are context numbers as the branches and numbers are associated with executing contexts (i.e., environments).

- select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (see page 268; note that the Bcc instruction will branch to the target instruction associated with the label when the comparison yields one result) and an instruction following the context branch instruction (the fall-through instruction, which inherently exists in conditional branching) based on the comparison (if the comparison yields that the numbers match, the target instruction will be selected; otherwise the following instruction will be selected).
- retrieve the selected other instruction. As is known, with a condition branch, either the target instruction or the subsequent instruction will be retrieved.

c) Steven has not explicitly taught a program counter for each executing context. However, this limitation is deemed to be inherent by the examiner. A program counter must exist because the program counter is the component that holds the address of the next instruction to be fetched. Without the PC, the system would not be able to fetch an instruction, which means that no work would be done. A program counter has to exist for each context because each context includes instructions that need to be fetched and executed.

20. Claims 2 and 4 are rejected under 35 U.S.C. 103(a) as being unpatentable over Saulsbury in view of Perets et al., U.S. Patent No. 6,564,316 (herein referred to as Perets).

21. Referring to claim 2, Saulsbury, as modified, has taught a computer program product as described in claim 1. Saulsbury has further taught that the branch instruction is of the format br=ctx[ctx#, label#], wherein the label# is a symbolic label corresponding to an address of the other instruction, wherein ctx# is the thread number, wherein the syntax "br=ctx" represents a branching operation based on ctx# matching the current thread number provided by the data processing apparatus. For the bz instruction, the context number (ctx#) is 0 and the branch destination is next1 (label#)". Again, see Fig.4. Saulsbury has not taught that the instruction format includes an optional_token field, wherein the syntax "optional_token" includes a value that causes the data processing apparatus to execute a number of instructions corresponding to the value of the optional_token following the context branch instruction before performing a branch operation. However, Perets has taught a branch instruction which includes an optional token. See Fig.6, step 600, and column 5, line 66, to column 6, line 5. The programmer-optional field (delay slot field) is used to specify the amount of delay slot instructions, where the use of delay slots for execution significantly speeds up the execution time of branch instructions. See column 5, lines 59-62. As a result, in order to speed up execution, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Saulsbury to include the optional_token field (delay slot field) of Perets.

22. Referring to claim 4, Saulsbury in view of Perets has taught a computer program product as described in claim 2. Saulsbury has further taught that the specified thread number has valid values of 0, 1, 2, or 3. See the bz instruction of Fig.4. Since the bz instruction is "branch if 0", 0 is a valid specified context number.

Response to Arguments

23. Applicant's arguments filed on January 22, 2008, have been fully considered but they are not persuasive.

24. Applicant argues the novelty/rejection of claim 1 on page 3 of the remarks, in substance that:

> "Earlier in the office action, the Examiner also stated that in Saulsbury, the current number may be "either dirty bit db0 or the flag which would be checked by the bz instruction." (Office Action, page 3). However, Saulsbury indicates that "the dirty bits currently stored by all of the dirty bit registers dbr- to dbrN-1 indicate which of the working registers wr0 to wrN-1 store operands that are to be saved to the main memory at the next context switch." (Col. 3, lines 30-33). Thus, Saulsbury describes at least one example in which the current number represented by db0 would not be a current executing thread number, even if the system were modified to support multithreading. The dirty bit registers store information related to the status of data, and it is therefore illogical to assume that the dirty bits would be thread numbers if the system were multithreaded, as the examiner appears to imply. Even if it were obvious to combine multithreading with the system described by Saulsbury (which Applicant does not concede), it would not be obvious to compare thread numbers in the manner recited in Applicant's claim 1."

25. These arguments are not found persuasive for the following reasons:

a) The examiner asserts that applicant is reading "thread number" much too narrowly. A "thread number", according to the examiner's broadest reasonable interpretation, is any number associated with a thread. A thread ID, a result of an addition instruction during the thread's execution, and a dirty bit indicating which register contents must be saved to memory at the next context/thread switch, are just a few of a plethora of numbers which may be considered thread numbers. If applicant desires "thread number" to mean something more narrow than just any general number associated with a thread in some manner, then the claim language must be further limited.

26. Applicant similarly argues the novelty/rejection of claim 1 under Steven on pages 3-4 of the remarks, in substance that:

> "Even if it were obvious to combine multithreading with the system described by Saulsbury (which Applicant does not concede), it does not follow that the register values and immediate values described by Steven would be thread numbers...
>
> In this example of Steven, the value zero is used for comparison with an operand. The value zero however is not a thread number in this instance. Nothing in the references teaches or suggests that the information in the registers could be a thread number, and thus it would not be obvious to compare thread numbers in the manner described in Applicant's claim 1, even if the system described by Steven were modified to be multithreaded."

27. These arguments are not found persuasive for the following reasons:

a) Again, the examiner asserts that applicant is reading "thread number" (or "current number" or "current executing thread number") much too narrowly. A "thread number", according to the examiner's broadest reasonable interpretation, is any number associated with a thread. A thread ID, a result of an addition instruction during the thread's execution, and a value of zero, which is used by the thread for branching purposes, are just a few of a plethora of numbers which may be considered thread numbers. If applicant desires "thread number" (and the like) to mean something more narrow than just any general number associated with a thread in some manner, then the claim language must be further limited.

Conclusion

28. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37

CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to DAVID J. HUISMAN whose telephone number is (571)272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/David J. Huisman/
Primary Examiner, Art Unit 2183
February 29, 2008